# Lab Four: MPC860 Caching and Performance

In this lab we explore the MPC860 caches and conduct an empirical study of cache performance. MPC860 has separate instruction and data caches. This lab focuses on the data cache. You need to read Chapter 7 in the MPC860 User Manual (UM); it is Chapter 8 if you use the hard copy.

The MPC860 data cache is a two-way associative cache that uses physical addresses. The cache has 16-byte line size. MPC860 allows programs to control caches explicitly, using the **mtspr** and **mfspr** instructions to read from or write to special purpose registers (SPRs). There are two sets of three registers: one for data cache and one for instruction cache. These registers are listed below. Read the chapters for detailed description.

| Register Name | SPR number | Description |
|---|---|---|
| DC_CST | 568 | Data cache control and status register |
| DC_ADR | 569 | Data cache address register |
| DC_DAT | 570 | Data cache data register |
| IC_CST | 560 | Instruction cache control and status register |
| IC_ADR | 561 | Instruction cache address register |
| IC_DAT | 562 | Instruction cache data register |

## 1 Caches and registers

You will experiment with the data cache and its control registers in this exercise. You will compare the memory access time when data cache is disabled versus the access time when it is enabled. The instruction cache should be enabled to minimize the impact of instruction fetching on results.

At the beginning of your code, follow the steps in the manual to initialize both instruction and data caches. You may enable only the instruction cache during the initialization and leave the data cache disabled. When sending commands to the cache control registers, use proper **sync** and **isync** instructions.

Before you load binary code into memory, issue the following two commands in the debugger to disable instruction cache and data cache:

```
rms IC_CST cmd 2
rms DC_CST cmd 4
```

You may put these commands into a file and use the `ex` command to invoke them.

### 1.1 Reading the time base

In the parts of the lab involving timing, you need access the MPC860 time base. Refer to the chapters you read in Lab 3 for enabling and reading it. Although the time base is a 64-bit value, the lower 32 bits (TBL) are enough for the timing purpose in this lab. Note that the clock rate to

the time base is about 1 MHZ. You are encouraged (but not required) to write a small program to measure the clock rate.

### 1.2 Timing a loop

Write a loop and test how much time it takes. You can read the time base before and after the loop, and the time it takes is the difference of two reads. Please note that the clock rate of MPC860 we are using is 20 MHZ and that most instructions complete in one or two cycles.

### 1.3 Reading with data cache disabled

The objective is to estimate the time it takes to load a word from memory. Try to make your measurement as accurate as possible. The overhead of the loop should not be included in memory access time.

### 1.4 Reading with data cache enabled

Repeat the experiment in the previous step with data cache enabled. All the memory accesses should be hits in this step. Compare the results in 1.3 and 1.4. Given a clock rate of 20 MHZ, how many cycles more does it take to load a word from main memory than from cache?

## 2 Testing the configuration of data cache

In this exercise, you will design schemes to test the parameters of a data cache system: the total data size, line size, and set associativity. The data cache should be enabled throughout the test, as well as the instruction cache. If you think that a program cannot test some of the parameters, explain why. Otherwise, write a program to test the parameters. Even if you already know the structure of the data cache on MPC860, you need to reach the same conclusions from the data you collected, and explain clearly how you draw the conclusions.

You can make reasonable assumptions about the parameters. For example, you can assume that the total cache size is between 1K bytes and 1M bytes. However, you cannot assume that the cache size is exactly 4K bytes.

You may discuss with other teams about the methods you are going to use, but do not copy other people's code.

You may implement your method with C language and test the cache configuration on your desktop (or laptop). This is not required though.

## 3 Deliverables

You will write a report adhering to the lab report requirements. Write separate objective, design, implementation, and results sections for exercise 1 and 2. You may be asked to demonstrate your programs. The report is due before the lecture on Oct 25, 2004.